グラフィクスとCAD 109－1
（2002. 11. 22）

3D CAD データを用いた組込み用ソフトウェア開発支援システムの構築－第一報 システムの概要と基本設計－

千田陽介† 橋間正芳† 佐藤裕一†

従来，組込み系の制御ソフトウェアの開発では，メカ試作機が出来上がるまで組込み用ソフトウェアの開発が進まないという問題があった．そこで，ソフトウェア開発において実際のメカに相当する部分を計算機上の仮想メカで代用する HIL (Hardware In the Loop) システムを構築した．HIL システムでは開発期間を短縮するだけでなく，実際のメカ試作機では実現が難しかった，任意のタイミングで発生する不具合を容易に再現したり，繰り返し実現することができ，ソフトウェアの品質を高めることができる．本稿では，HIL システムの技術ポイントと試作結果を紹介するとともに，本システムを用いた効率的な組込みシステムの設計手法について提案するものである．

An Embedded-Software Development System using 3D-CAD Data
– 1st report : Basic System of HIL Simulation–

Yosuke Senta,† Masayoshi Hashima† and Yuichi Sato†

In this paper, we describe a "HIL (Hardware In the Loop)" simulation system constructed on a mechanical simulator. In the HIL simulation system, a complete mechanical model, virtually constructed on a computer, is controlled by an actual electric board. Therefore, engineers can develop its control software based on this virtual model before an actual model is completed. The character of this virtual model can be easily changed by tuning mechanical parameters of the model, so that engineers can effectively predict an error tolerance of the whole system. This system leads to quite efficient concurrent development of mechanical model and embedded software.

1. はじめに

現在，製造業全般で製品開発の期間短縮・コスト削減が急務となっており，その対応として，3 次元 CAD を利用したデジタルエンジニアリングが普及してきている．しかし，デジタルエンジニアリングの普及によって製品開発のサイクルが短くなるに従い，組込み用ソフトウェア (ファームウェア: 以下 F/W と略す) 開発者への負担は大きくなってきている．これは，F/W は基本的に実際のメカ試作機を用いて開発や検証を行う必要があるため，試作機完成前の段階で効果の高いデジタルエンジニアリングの恩恵を授かることができなかったためである．

そこで今回，製品の設計時に出てくる 3 次元 CAD データや設計仕様書を基に，計算機上に実物そっくりに動くバーチャルなメカ (仮想メカ) を構築し，実機の代わり仮想メカを用いて F/W を開発する HIL (Hardware In the Loop) シミュレーションシステムを構築した[1]．このシステムは，仮想メカを実機とみなして F/W を開発するため，試作機の完成を待たずに開発を進めることができる．さらに，センサやモータの不具合，異物が噛みこんでメカが動かなくなった等の現象を簡単に再現することができるため，様々な異常状態に対する F/W 動きを十分に検証することができる．

本稿では今回構築した HIL システムの試作結果と，本システムを用いた新しい製品の設計手法について述べていく．

† 富士通研究所
Fujitsu Laboratories LTD.

2. HIL システム

2.1 システム概要

本システムの概要を **図 1** ～ **図 3** に示す．従来は，図 1 のように実際の試作機を用いて制御基板上の F/W を開発していたのに対し，本システムでは図 2 のように試作機の部分を計算機上の仮想メカに置き換

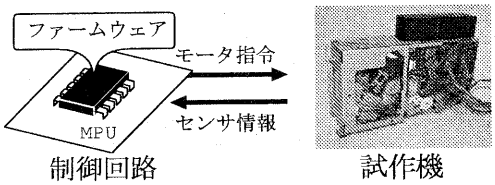

図 1 従来のソフトウェア開発環境
Fig. 1 Combined test using actual mechanism

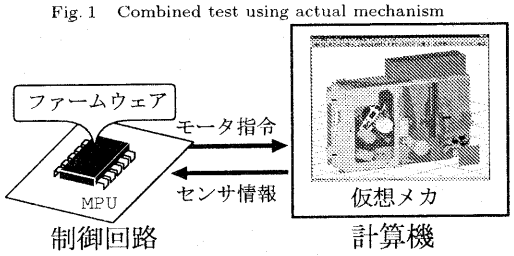

図 2 HIL によるソフトウェア開発環境
Fig. 2 Overview of HIL simulation system

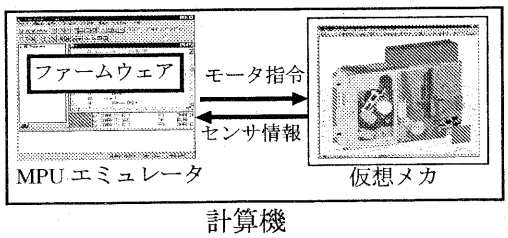

図 3 HIL システムの発展形
Fig. 3 HIL simulation using firmware emulator

えて F/W を開発する．さらにこの考えを発展させ，図 3 のように，実際の制御ボードの代わりに MPU のエミュレータや状態遷移シミュレータなどの仮想的な F/W 実行環境を用いることで，メカも回路もない完全な設計段階から F/W 開発を始めることもできる．

2.2 従来の HIL システムとの違い

実機の代わりに計算機上の仮想メカを用いて F/W の開発を行う HIL シミュレーションは，既に自動車の ABS やエンジン[3]，ミサイル[4]，磁気ディスク[2] 等の制御 F/W 開発で実現されている．しかし，これらの F/W 開発は，主にサーボ制御が主眼となっている．一般にサーボ制御の対象は，もともとモデル化しやすいように設計されており，数式もしくはブロック図の形で表すことができる．そこでこれらの HIL システムの仮想メカは，計算機上でこれらの数式を解いたり，ブロック図を演算することで実現していた．

一方今回，筆者らは，タスク制御の F/W を開発するための HIL システムを開発した．制御対象には，プリンタやカーオーディオデッキ (CD/MD チェンジャ

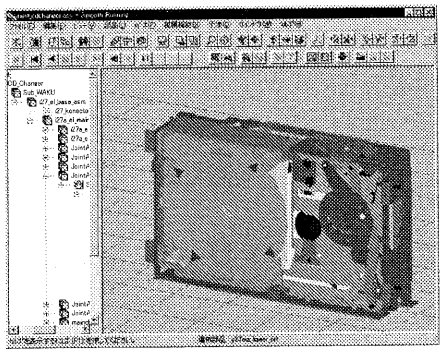

図 4 VPS 外見
Fig. 4 Overview of VPS

等) などがある．これらの機器は，複雑な動きが必要な反面，コンパクトに収まっていることが求められており，ギアやカムが入り組んだ複雑な機構になっている．しかし，制御自体はステッピングモータや単純な DC モータ，on/off 判定しか行わないセンサ等を用いており，動的挙動の厳密性は必要ない．このため，タスク制御用の仮想メカは，従来のサーボ制御用 HIL システムのように，数式やブロック図で実現することは難しく，またその必要もない．そこで，筆者らはタスク制御用の仮想メカの動作環境として，VPS を用いることにした．

3. VPS

3.1 目　　的

VPS (Virtual Product Simulator) とは筆者らが開発している 3 次元リアルタイムシミュレータで，設計段階で製品がどのような動きをするのか計算機上で確認することを主な目的としている．VPS の外見を**図 4** に示す．

VPS では，一般の 3 次元 CAD からコンバータを介してポリゴンデータとアセンブリデータを取り込み，回転と並進の関節，ギア，カム (溝機構を含む)，クラッチなどの汎用的な機構定義を設定することで，ほとんどの機構を簡単に設定することができる．機構設定されたモデルは，マウスのドラッグ操作によって任意の部品を動かすことができ，それにより関連した部品がどのように動くのかをリアルタイムに確認することができる．

これにより，従来試作品が完成するまでは設計者の頭の中にしかなかった製品イメージを多人数で共有し，部品間の干渉や機構全体の動き，操作性，視認性，保守性，組立性[5] など，あらゆる観点から製品を事前に評

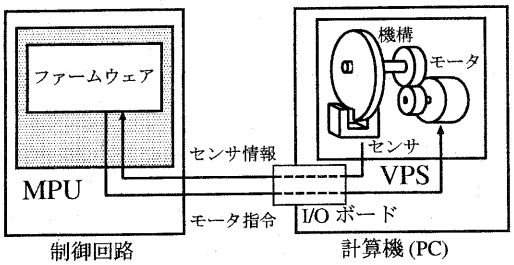

図 5 HIL シミュレーションの基本原理
Fig. 5 A principle of HIL system

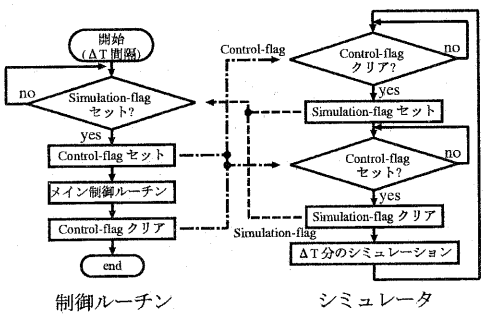

図 6 同期アルゴリズムのフローチャート
Fig. 6 Flowchart of a synchronized algorithm

価することができる.

また, デジタルデータを多くの部門で活用するため, LCA (Life Cycle Assessment) や解体性の評価機能, 公差解析機能, ネットワーク間で繋がった遠隔地間の人々がある製品モデルに対し同時に操作し議論する機能など様々な機能を開発しており, 本 HIL シミュレーションシステムもその一つである.

3.2 高速シミュレーション技術

HIL シミュレーションを行うためには, 製品の動きを高速に計算し描画する必要がある. VPS ではギアやカム等の関節間の連動関係をギア比やテーブルの形で保存しており, シミュレーション実行時はこれらの値を参照することで, 高速な機構演算を実現している. このように力学的な観点から機構解析を行っていないため, 設定によっては現実にはありえない動きや, 摩擦やたわみ等が原因で不可能な動きでも, VPS 上では動作するように見える恐れはある. しかし設計者個人が思い描いたメカニズムを共有する目的では支障ない.

さらに, Occlusion Culling や Level of Detail などの高速描画技術を利用しており, 大規模なモデルでも実用的な速度で表示することができる.

3.3 異常状態

VPS は設計者が描いた理想的な製品メカを表現することを主目的としているが, より現実に近い動きを表現するため, ギアやカム機構にヒステリシスを持たせてギアのバックラッシや溝機構のガタなども再現できる. さらに, 高速干渉チェック機能[6] を利用して異物混入や部品の引っかかりなどの状態を作り出すこともできる.

4. VPS による HIL シミュレーション

4.1 基本原理

VPS を用いた HIL シミュレーションの基本原理を **図 5** に示す. まず, I/O ボードの入力信号に応じて VPS モデル内のモータに相当する部品の姿勢が変化する. すると, 機構演算機能によって関連する部品が動き, センサに相当する部品の姿勢も変化する. さらにセンサに相当する部品の姿勢に応じ I/O ボードへの出力が変化する. これらの I/O ボードの入出力信号と F/W が動作している MPU チップの I/O ピンとを接続することで HIL シミュレーションが実現する.

このような I/O ボードを使う方法は **図 2** で示した実ボードを使った HIL シミュレーションの実現例である. 図 3 のように F/W エミュレータを用いる場合は I/O ボードの代わりにソケット通信や, MS-Windows のメッセージ通信を用いて信号を送受信している.

4.2 同期アルゴリズム

以上のように HIL シミュレーションの基本原理は非常に単純である. しかし実際には 3D 描画や干渉チェックなど時間のかかる処理があり, シミュレーションのサンプリング間隔を充分長くしなければ, 実機と同じ速度でシミュレーションすることができない. ところが, サンプリング間隔が長いとシミュレーション精度が悪くなってしまう. この問題を解決するには以下の二つの方法がある.

(1) 干渉チェックなど時間のかかる機能の利用を諦め, 同時に描画の間隔を間引くことでリアルタイム性を確保する
(2) 逆にリアルタイム性を諦め, F/W の実行速度をシミュレーション速度まで遅らせる

今回構築したシステムは, ユーザ指定でいずれかの方法を選択できるようにした. ここでは, (2) の F/W の実行速度をシミュレーション速度まで遅らせ, 両者の速度を一致させる同期アルゴリズムについて述べる.

同期アルゴリズムは, 一般に F/W の制御ルーチンが割込みやタイマのポーリングまたはマルチタスク OS の機能などによって, 一定周期で呼び出されていることを利用している. **図 6**, **図 7** に同期アルゴリズムの原理を示す. 同期アルゴリズムは, モータや

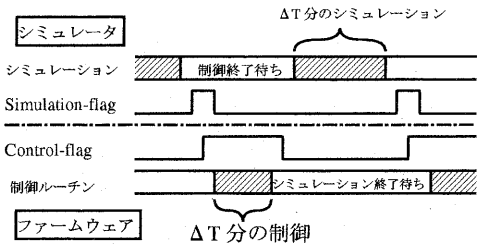

図7 図6に示したアルゴリズムのタイミングチャート
Fig. 7 Timming chart of **Fig.6**'s algorithm

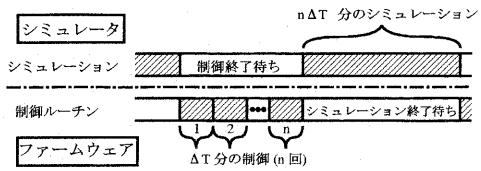

図8 同期アルゴリズムの高速化
Fig. 8 Timming chart for fast simulation

センサ信号の他に同期のための信号 (Simulation-flag, Control-flag) を設け, F/W とシミュレータが互いに現在の状態を知らせ合うことで制御ルーチンとシミュレーションルーチンを交互に実行する. この際, シミュレーション上のサンプリング間隔を制御ルーチンが呼び出される周期と一致させると時間的に厳密なシミュレーションを行うことができる.

このような同期アルゴリズムを実現するためには F/W の中に同期のための処理を加える必要があるが, その変更量はわずかなので特に問題になることはない.

4.3 同期アルゴリズムの高速化

前節で述べた同期アルゴリズムは, F/W の制御周期 ΔT と同じ間隔でシミュレーションしなければならない. そのため不必要に精度が高くシミュレーションが遅くなる恐れがある. そこで **図8** のように, シミュレーションは ΔT を n 倍 (n : 整数) した $n\Delta T$ 間隔で行い, F/W は n 回連続して制御ルーチンを実行することで高速にシミュレーションを行うことも可能にした. この整数 n はユーザが直接指定できる他, 普段は速度優先で n を大きく取ってシミュレーションを行い, 干渉が発生/センサ値が切り替わる可能性が高いなど, 精度が必要な時だけ n を小さくしてシミュレーションを行うこともできる.

4.4 モータ, センサモデル

今回, モータとして DC モータとステッピングモータを用意した. このうち DC モータは, 定電圧で定負荷を駆動することを想定し, 簡単な一次遅れ系でモデリングした. このため, 定電流で駆動する場合や, カムやクラッチ等によって駆動中にモータの負荷が変化する機構の場合, モータの動きを正確に再現しないことになる. しかし, タスク制御の検証では厳密な力学的挙動よりも, 高速にシミュレーションできることや, 簡単に設定できることの方が優先されるため, 単純な一次遅れ系のモデルで十分だと考えている.

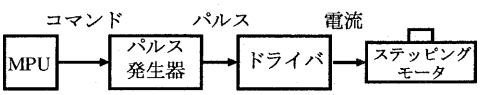

図9 ステッピングモータの一般的な駆動回路の構成
Fig. 9 A driving circuit model for stepping motor

一次遅れ系において, 時間 $t+\Delta t$ におけるモータ速度 ω_k は, 時間 t におけるモータ速度 ω_{k-1} を用いて次のように表すことができる.

$$\omega_k = \omega_{k-1} + (\omega_i - \omega_{k-1})(1 - e^{-\frac{\Delta t}{T}}) \qquad (1)$$

となる. ここで ω_i は指令速度, T は定数である. これより, Δt 間にモータが回転する角度 $\Delta\theta$ は式 1 を積分して,

$$\begin{aligned}\Delta\theta &= \int_0^{\Delta t} \omega_{k-1} + (\omega_i - \omega_{k-1})(1 - e^{-\frac{t}{T}})dt \\ &= \omega_{k-1}\Delta t + (\omega_i - \omega_{k-1})\{\Delta t - T(1 - e^{-\frac{\Delta t}{T}})\}\end{aligned} \qquad (2)$$

となる.

本システムでは, ユーザが入力信号パターンに対する目標速度 (ω_i) と, 目標速度で安定するまでの整定時間 ($4T$) を設定する. シミュレーション中はこれらの値とともに, 前サンプル時のモータ速度 (ω_{k-1}) と角度 (θ) から, 式 1, 2 を用いて, モータの速度と角度を更新することで一次遅れ系を表現している.

一方ステッピングモータは, **図9** のように MPU 外部にパルス発生器を備えた回路構成で用いられることが多い. この構成では MPU (F/W) はコマンドを発行するのみで, 後はパルス発生器が自動的にパルスを発生する. そのため, 例えばセンサが切り替わるとモータが止まる F/W に対し同期アルゴリズムを用いて検証しようとした場合, パルス発生器が本来センサが切り替わるだけのパルスを発生しても, 同期アルゴリズムによって F/W は停止しているため, センサが切り替わったことを検知することができず, シミュレーションが破綻してしまう.

そこで, ステッピングモータの実装は

(1) 4.2 節 (1) で述べたリアルタイム性を確保した

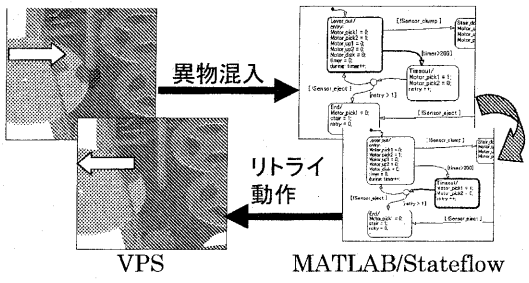

図 10 HIL シミュレーション実行例
Fig. 10 Sample of HIL simulation

シミュレーション.

(2) 図3のような, エミュレータを用いたシミュレーション

に限定することにした.

(1) のリアルタイム性を確保したシミュレーションでは, シミュレーションのサンプリング間隔の間に変化したパルスの挙動を専用ボードを用いてメモリ空間にバッファリングし, それを解析することでモータの動きを得ることができる. また, (2) のエミュレータを用いたシミュレーションでは, パルス発生器への指令コマンドを直接シミュレータに取り込むことで実現できる. コマンドに対するモータの動きは, 台形や S 字等の代表的な加減速パターンの他, ユーザ定義によって任意の加減速パターンを再現可能にした. これにより, 市場に出ているほとんどのパルス発生器に対応することができる.

また, センサとして機械的, 光学的な on/off スイッチ, エンコーダ, ポテンショメータを用意した. これらは, シミュレーションにおいて, センサに相当する部品の姿勢に応じた出力をする.

4.5 モータやセンサの不良モデル

タスク制御の F/W では一連の動作ができることはもちろん, 部品が噛み込んで動かなくなった, モータやセンサが故障した動作など, 異常状態における動作が重要な位置を占める. そのため, モータやセンサの不良状態を簡単に作り出すことができるようにしている. モータやセンサの不良状態や 3.3 節で述べたメカの不良状態を用いて様々な故障の組み合わせを発生させ, 異常状態におけるフェールセーフやリトライ動作の検証を行うことができる.

モータの不良は, 接触不良や断線, ドライバの故障を想定し, 動作命令を受けてもモータが回転しないことで表現している. また, 前節の DC モータの一次遅れの整定時間を大きくとることも, モータ不良状態の一つと考えることができる.

同様にセンサの不良も, 断線やアンプの故障を想定し, センサに相当する部品の位置に関わらず on や off, 0V や Vcc で固定することができるようにしている. さらに on/off スイッチではチャタリングの発生機能も備えた. チャタリングの設定パラメータは, スイッチが切り替わった瞬間からチャタリングが発生する時間と on/off を間違える確率の二つで, これらを on から off への変化, off から on への変化について個別に設定可能にしている. ここでチャタリングを確率で表したのは, 一般にチャタリングの周期が制御周期よりも速く, F/W にとってサンプリング毎に on なのか off なのかは不確定だからである.

5. 試作結果

5.1 適応例

本システムを実際の CD チェンジャに適用し評価を行った. 評価に用いたモデルは, 約 30 個のギア・カム, 3 個のモータ, 4 個のセンサから構成され, 約 200 部品からなる.

まず初めに, 状態遷移図シミュレータとして MATLAB/Stateflow を用いて HIL シミュレーションを行った. **図 10** に HIL シミュレーションの実行例を示す. オペレータが VPS 上で仮想的にドライバを差し込み, 機構の動きを強引に止めた時, フェールセーフ処理の実行とメカのリトライ動作が連動している様子を視覚的に把握することができた.

続いて市販されている製品用 F/W に同期処理を追加し, 制御ボードとパソコンを結合した. このインタフェースは実際と同じ配線で, PIO ボードと DA ボートを介してパソコンに接続した. 結合試験では、10ms のサンプリング間隔で実行し, 実物の 4 倍程度の速度でディスク交換の様子を再現することができた.

5.2 VPS を用いた新しい製品開発手法

筆者らは VPS を用いることで効率的に製品を開発することができると考えている. **図 11** にその概要を示す. まず, メカ開発においては, 3 次元 CAD により設計を進め, VPS で仮想メカを構築し機構の検証を行う. 一方 F/W 開発では, この仮想メカを用いてタスク制御の F/W を状態遷移ベースで設計していく. 続いて状態遷移図から自動的に生成した C コードを実装レベルにチューニングし, 制御ボートによる HIL シミュレーションを行う. この間にメカの方で設計変更が発生しても変更結果を簡単に仮想メカに反映することができるため, 即座に F/W の対応が可能である. 最後に組上がった試作機と F/W を含む制御ボートとを初めて結合して動作試験を行い, 製品を完成させる.

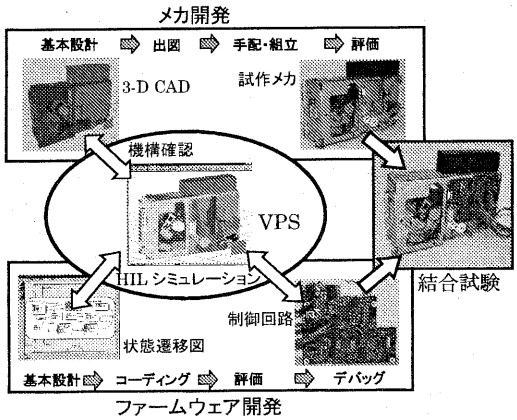

図 11 VPS による新しい設計手法
Fig. 11 Development process based on VPS

以上のように試作メカが完成する前の段階で大部分の動作を確認しておくことができるため, 最終工程の結合試験で見つかる不具合は, 本システムを用いなかった場合に比べ大幅に減少し, 試作回数の削減が可能となる. また, F/W とメカを並行して開発することができるため開発期間の短縮も期待できる. 残念ながら詳細は述べられないものの, 従来の半分近く工数削減が実現できた現場もある.

6. おわりに

タスク制御のファームウェア開発にあたり実際のメカに相当する部分を 3 次元機構シミュレータ VPS で代用する HIL シミュレーションシステムを開発した. 本システムを用いることで一通りのコードチェックや異常状態の処理の確認を実機の完成前に行うことができ, 実機を用いたデバッグ工数を減らすことが可能となった.

現在本システムは, 事務機や自動販売機などへの適用を拡大しており, 今後, 回路設計シミュレータとの連携を進め, メカ, ソフト, ハードを結合した協調検証シミュレータを構築する予定である.

なお本システムは “VPS/IOConnector” として製品化されている[7].

謝辞 本研究への貴重なアドバイスとモデルを提供して下さった富士通テン (株) AVC 本部コンポーネント事業部デッキ技術部の関係者各位に感謝致します.

参考文献

1) M.Hashima et.al.: Design and Manufacturing Methodology for Mechanical System Using Virtual Product Simulator, CD-ROM Proc. of the ASME Int. DETC/CIE (2002)
2) 千田ほか: 磁気ディスクのサーボプログラム開発を支援する HIL シミュレーションシステム, 電子情報通信学会技術研究報告, 信学技法, Vol.100 No.504, pp.1–6 (2000)
3) H.Hanselmann, Hardware-in-the-Loop Simulation Testing and its Intefration into a CACSD Toolset, IEEE Int. Symp. Computer-Aided Control System Design, 152/156 (1996)
4) P.E.PACE et,al., Effectiveness Calculations in Captive-Carry HIL Missile Simulator Experiments, IEEE Trans. Aerospace and Electronic Systems, vol.34, No.1, 124/136 (1998)
5) 平田ほか: Virtual Production Simulator –自動アセンブリアルゴリズムの開発–, 第 15 回ロボット学会学術講演会予稿集, pp.1039–1040 (1997),
6) 佐藤ほか: 高速干渉チェックアルゴリズム Contact Scope の開発と応用, 第 13 回ロボット学会学術講演会予稿集, pp.373–374 (1995),
7) http://www.nagano.fujitsu.com/fjvps